PLAN:

- ✓ Introduction
- ✓ Les variables
- ✓ Les commandes
- ✓ Scripts Shell
- ✓ Structures de contrôle
- ✓ Substitution
- ✓ Conclusion

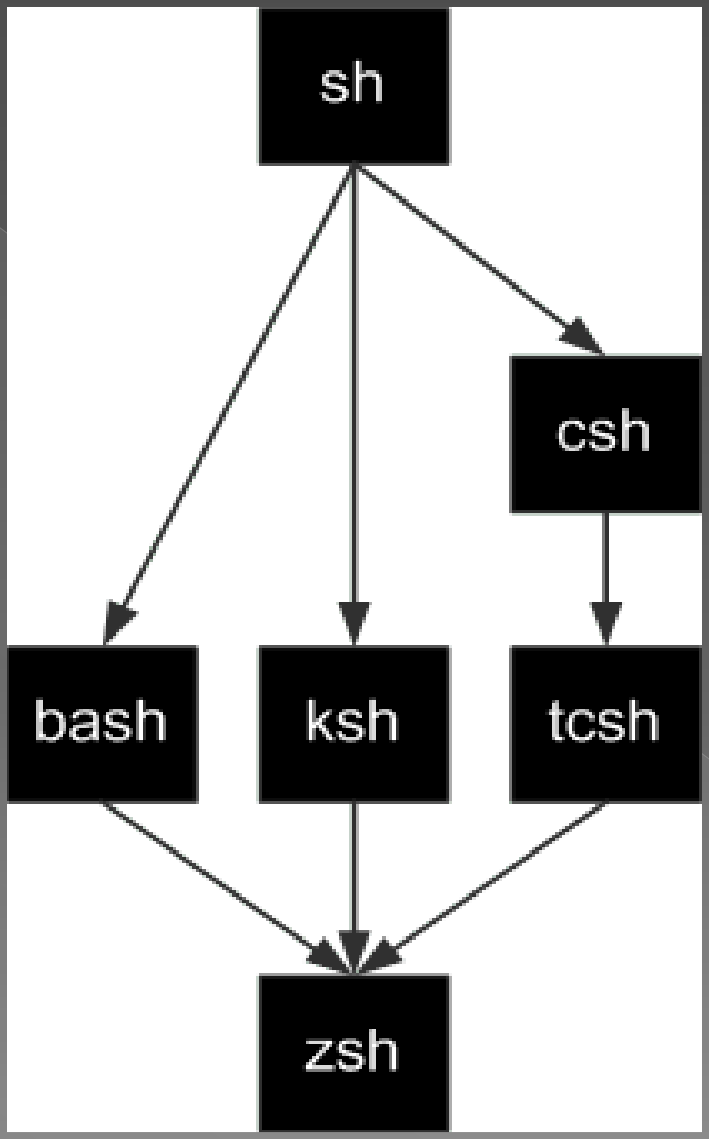
INTRODUCTION

Définition:

- Interface humain-ordinateur (IHO) en ligne de commande.

- Le Shell a un double rôles :
  - Interpréteur de commandes
  - Langage de programmation

II. Historique:



- **csh :** C Shell. Un shell utilisant une syntaxe proche du language C
- **sh :** Bourne Shell. L'ancêtre de tous les shells.
- **bash :** Une amélioration du Bourne Shell.

echo

Affichage de texte sur la sortie standard

Exemple :

```
najihi@ubuntu:~$ echo salut tout le monde
salut tout le monde
```

Affichage d'une variable

Exemple :

```
najihi@ubuntu:~$ message="Bonjour tout le monde"
najihi@ubuntu:~$ echo $message
Bonjour tout le monde
```

Quotes

Il existe trois types de quotes :

- les apostrophes ' ' **(simples quotes)** .
- les guillemets " " **(doubles quotes)** .
- les accents graves ``` **(back quotes)** : substitution commande

◉ Les simples quotes ' ‘

```
najihi@ubuntu:~$ message='Bonjour tout le monde'
najihi@ubuntu:~$ echo 'le message est: $message'
le message est: $message
```

◉ Les doubles quotes " "

```
najihi@ubuntu:~$ message='Bonjour tout le monde'
najihi@ubuntu:~$ echo "Le message est: $message"
Le message est: Bonjour tout le monde
```

Les variables

- Les variables simples
- Les tableaux
- Les variables d'environnement

Les variables simples

Syntaxe :

variable=chaîne

Exemple :

```
najihi@ubuntu:~$ nom=pierre          #Affectation de "Pierre" à la variable "nom"
najihi@ubuntu:~$ objet=voiture       #Affectation de "voiture" à la variable "objet"
najihi@ubuntu:~$ coul=blue           #Affectation de "blue" à la variable "coul"
najihi@ubuntu:~$ txt="$nom a une $objet $coul"   #Melange de variables
najihi@ubuntu:~$ echo $txt           #Attention à ne pas oublier le caractère "$"
pierre a une voiture blue
```

◉ Les tableaux

Syntaxe :

tableau=(chaîne1 chaîne2 ...)

${tableau[2]} : Affichage de la case N°2

${tableau[*]} : Affichage de toutes les cases

Exemple :

```
najihi@ubuntu:~$ tableau=(chaine1 chaine2 chaine3)
najihi@ubuntu:~$ echo ${tableau[2]}   #Affichage du contenu de la case 2
chaine3
najihi@ubuntu:~$ echo ${tableau[*]}   #Affichage de l'ensemble du contenu du tableau
chaine1 chaine2 chaine3
najihi@ubuntu:~$ echo ${tableau[@]}   #Affichage de l'ensemble du contenu du tableau
chaine1 chaine2 chaine3
```

◉ Variables d'environnement

HOME

```
najihi@ubuntu:~$ echo $HOME
/home/najihi
```

USER

```
najihi@ubuntu:~$ echo $USER
najihi
```

PWD

```
najihi@ubuntu:~$ echo $PWD
/home/najihi
```

SHELL

```
najihi@ubuntu:~$ echo $SHELL
/bin/bash
```

PATH

```
najihi@ubuntu:~$ echo $PATH
/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/
usr/games
```

HOSTNAME

```
najihi@ubuntu:~$ echo $HOSTNAME
ubuntu
```

HISTSIZE

```
najihi@ubuntu:~$ echo $HISTSIZE
1000
```

COMMANDES SHELL

C'est quoi une commande Shell

Nom de la commande -options arguments

TYPES DE COMMANDES

TYPES DE COMMANDES

- ✓COMMANDES INTERNES
- ✓COMMANDES EXTERNES
- ✓COMMANDES COMPLEXES
- ✓FICHIERS DE COMMANDES

COMMANDES INTERNES:

ATTENTION : Selon le shell utilisé les commandes internes peuvent etre différentes

```
pubuntu@pubuntu:~$ type cd
cd is a shell builtin
```

COMMANDES EXTERNES:

EXEMPLE:

```
pubuntu@pubuntu:~$ type cp
cp is /bin/cp
```

COMMANDES COMPLEXES
FICHIERS DE COMMANDES

Commandes essentielles

read

Syntaxe :

$ read nom_variable [...]

Options courantes:

Pas d'option.

Exemple :

```
pubuntu@pubuntu:~$ read a b
c'est un exemple
pubuntu@pubuntu:~$ echo $a
c'est
pubuntu@pubuntu:~$ echo $b
un exemple
```

expr

Syntaxe :

expr argument1 opérateur argument2

Options courantes:

Pas d'option.

Exemple :

```
pubuntu@pubuntu:~$ a=2
pubuntu@pubuntu:~$ b=3
pubuntu@pubuntu:~$ a=`expr $b % $a`
pubuntu@pubuntu:~$ echo $a
1
```

Remarques

- *, >, >=, < et <=.
- ordre prédéfini de priorité des opérateurs.

set

Syntaxe :

set *nom_var* active la variable

set affiche la liste des variables actives

set param1 param2 param3

Option courante

-u refuser les variables indéfinies

-a exporter toutes les variables

Exemple

```
pubuntu@pubuntu:~$ set poire fraise
pubuntu@pubuntu:~$ echo "Nombre de  paramètres sont : $#"
Nombre de  paramètres sont : 2
pubuntu@pubuntu:~$ echo "Les paramètres sont : $@"
Les paramètres sont : poire fraise
```

exit :

Syntaxe :

exit

Options courante :

Pas d'option.

shift :

Syntaxe:

shift [n]

Options courantes:

Pas d'option.

Exemple :

```
pubuntu@pubuntu:~$ set ab cd
pubuntu@pubuntu:~$ echo $1
ab
pubuntu@pubuntu:~$ shift
pubuntu@pubuntu:~$ echo $1
cd
pubuntu@pubuntu:~$ shift
pubuntu@pubuntu:~$ echo $1
```

Scripts Shell

- Deux moyens de programmation :

=> Direct

=> Scripts

C'est quoi un script

Un script est un suite de commandes écrite dans un fichier.

Comment écrire nos scripts

Trois étapes suffisent :

- Création du fichier
- Rendre le fichier exécutable
- Exécution du script

=> Création du fichier :

- ***Indication du nom du Shell***
- *Exécution des commandes*

syntaxe particulière :

Installation du Shell :

```
# apt-get install csh
$ chsh
```

=> Création du fichier :

- *Indication du nom du Shell*
- ***Exécution des commandes***

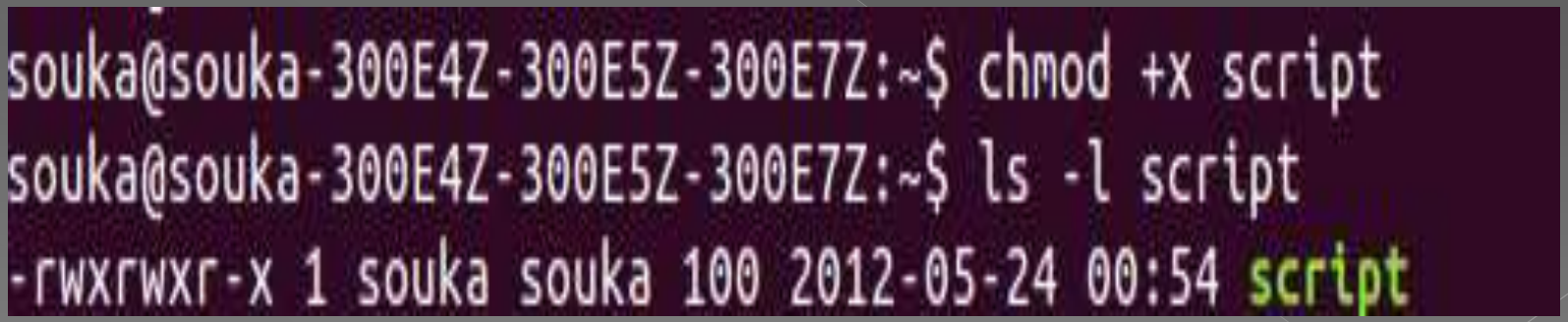
Exemple:

```
#!/bin/bash

pwd     # Affichage du répertoire courant

ls      # Affichage de la liste des fichiers
```

Rendre le fichier exécutable

« Exécutable »

chmod +x



Exécution du script

1- « ./ nom du script »

Exemple:

```
souka@souka-300E4Z-300E5Z-300E7Z:~$ ./script
/home/souka
Bureau        examples.desktop  images6.tif  script          Ubuntu One
cours linux   expo              Modèles      sOukA           Vidéos
default.jpg   Images            Musique      Téléchargements
Documents     images34.tif      Public       Travaux dirigés
```

2- « echo $PATH »

```
souka@souka-300E4Z-300E5Z-300E7Z:~$ echo $PATH
/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin
:/bin:/usr/games
```

Exemple :

```
souka@souka-300E4Z-300E5Z-300E7Z:~$ sudo su
root@souka-300E4Z-300E5Z-300E7Z:/home/souka# mv script /usr/bin
root@souka-300E4Z-300E5Z-300E7Z:/home/souka# exit
exit
souka@souka-300E4Z-300E5Z-300E7Z:~$ script
/home/souka
Bureau       examples.desktop  images6.tif  sOukA           Ubuntu One
cours linux  expo              Modèles      Téléchargements  Vidéos
default.jpg  Images            Musique      Travaux dirigés
Documents    images34.tif      Public       typescript
```

Les srtuctures de contrôles

- LES STRUCTURES CONDITIONNELLES:
  - La structure if
  - La structure case
- LES BOUCLES
  - La structure while until
  - La structure for

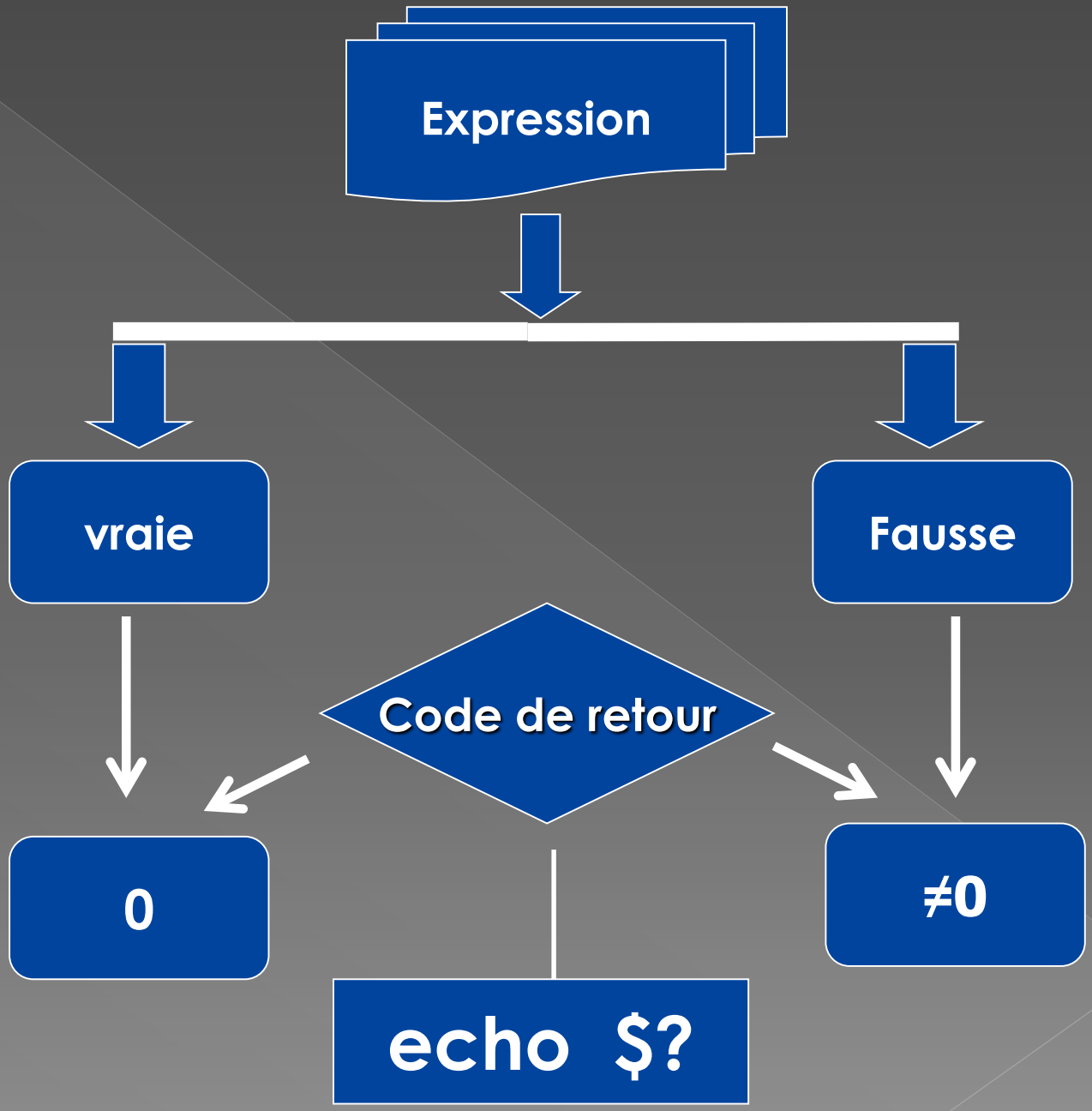
remarque IMPORTANT

- IL convient de préciser que chaque shell à sa propre syntaxe .
- Le shell par défaut est bash

La commande test

1.qu'est ce qu'un test

- ***opération dont le but est d'évaluer la valeur d'une expression***


Expression
vraie
Fausse
Code de retour
0
≠0
echo $?

Syntaxe

- *Ces deux syntaxes sont équivalents:*
  - *test expression*
  - *[expression]*

```
administrateur@ubuntu:~$ [ 2=2 ]
administrateur@ubuntu:~$ echo $?
0
administrateur@ubuntu:~$ test 2=2
administrateur@ubuntu:~$ echo $?
0
```

- *on ne doit pas écrire [expression] mais [↔ expression ↔]*

  ↔:espace

```
administrateur@ubuntu:~$ [2=2]
[2=2]: command not found
administrateur@ubuntu:~$ echo $?
127
```

ContrôleurS de test

en bash il est possible d'affecter trois types test différents :

- ***Des tests sur des chaînes de caractères***
- ***Des tests sur des nombres***
- ***Des tests sur des fichiers***

Tests sur chaînes de caractères

Condition	Signification
$chaine1 = $chaine2	Vérifie si les deux chaînes sont identiques. Notez que bash est sensible à la casse : # b # est donc différent de # B #. Il est aussi possible d'écrire # == # pour les habitués du langage C.
$chaine1 != $chaine2	Vérifie si les deux chaînes sont différentes.
-z $chaine	Vérifie si la chaîne est vide.
-n $chaine <=> ! -z$chaine	Vérifie si la chaîne est non vide.

Exemple :

```
#!/bin/bash
if [ ! -z $1 ]
then
echo "chaîne non vide"
else
echo "chaîne vide"
fi
~
~
```

Exécution du script :

```
souka@souka-300E4Z-300E5Z-300E7Z:~$ ./chaine soukaina
chaîne non vide
souka@souka-300E4Z-300E5Z-300E7Z:~$ ./chaine
chaîne vide
```

Tests sur des nombres

Opérateur arithmétique	signification	signe
eq	Equal	=
-ne	Not equal	!=
-gt	Greater than	>
-lt	Lesser than	>
-ge	Greater or equal	>=
-le	Lesser or aqual	<=

Exemple :

```
#!/bin/bash
if [ $1 -lt 0 ]
then
echo "le nombre est négatif"
else
echo "le nombre est positif"
fi
```

Exécution du script :

```
souka@souka-300E4Z-300E5Z-300E7Z:~$ ./nombre 3
le nombre est positif
souka@souka-300E4Z-300E5Z-300E7Z:~$ ./nombre -8
le nombre est négatif
```

Tests sur des fichiers

Condition	Signification
-e $nomfichier	Vérifie si le fichier existe.
-d $nomfichier	Vérifie si le fichier est un répertoire. N'oubliez pas que sous Linux, tout est considéré comme un fichier, même un répertoire !
-f $nomfichier	Vérifie si le fichier est un vrai fichier cette fois, pas un dossier.
-L $nomfichier	Vérifie si le fichier est un lien symbolique (raccourci).
-r $nomfichier	Vérifie si le fichier est lisible (r).
-w $nomfichier	Vérifie si le fichier est modifiable (w).
-x $nomfichier	Vérifie si le fichier est exécutable (x).
$fichier1 -nt $fichier2	Vérifie si fichier1 est plus récent que Fichier2 (newer than).
$fichier1 -ot $fichier2	Vérifie si fichier1 est plus vieux que Fichier2 (older than).

I. La structure if

- **Syntaxe:(Plus générale)**

```
if test condition_1  # ou if [ condition_1]
then
commandes1
[ elif test condition_2 ; then commandes ]…
[ else   commandes3 ]
 fi   # l'envers de if (fin de bloc)
```

II. La structure case

(un choix parmi multiples choix)

Syntaxe:

```
case  $variable  in
     Valeur1)
    Commande1;;
     Valeur2)
    Commande2;;
   ……..
     *)      # tous les autres cas
   Commande;;
esac # (l'envers de case) ( Fin de bloc case )
```

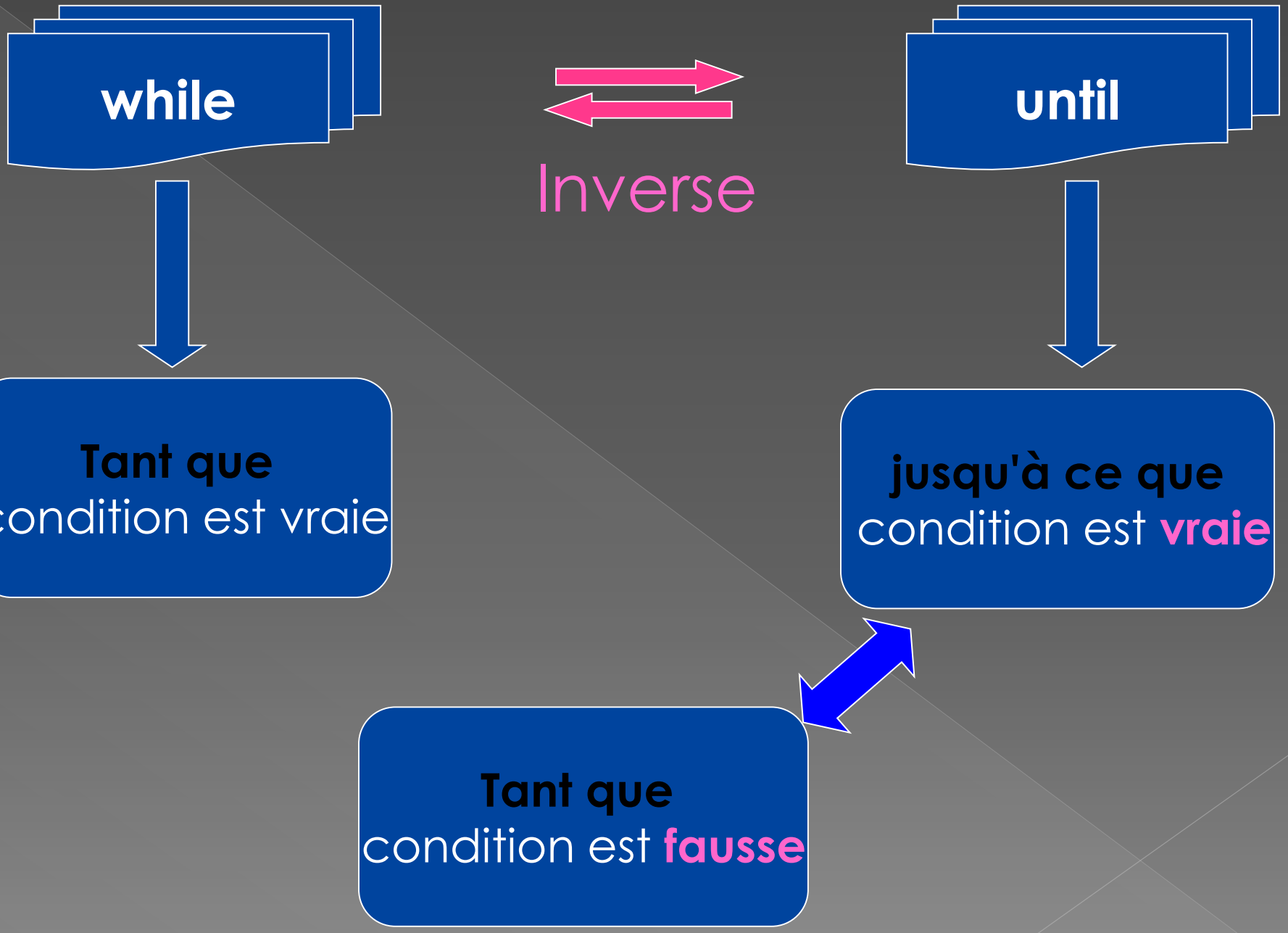
Comment fonctionne

while et until

Syntaxe:

```
while condition
do
  commandes
done
```

```
until condition
do
  commandes
done
```


while
Inverse
until
Tant que
condition est vraie
jusqu'à ce que
condition est vraie
Tant que
condition est fausse

IV. La structure for

- Parcourir une liste de valeurs définies.

Syntaxe:

```
for variable in liste-de-valeurs
do
    commandes
done
```

Exemple

*groupe

```
#!/bin/bash
echo "les membres de mon groupe sont : "
for nom in 'Soukaina Boujadi' 'Soukaina Najihi' 'Soukaina Ajankar'
'Meryem Abounasr'
do
echo " $nom "
done
```

```
administrateur@ubuntu:~$ ./groupe
les membres de mon groupe sont :
 Soukaina Boujadi
 Soukaina Najihi
 Soukaina Ajankar
 Meryem Abounasr
administrateur@ubuntu:~$
```

Les paramètres positionnelles

- $#
- $0
- $1, $2, $3 ... $9
- $* et $@

Exemple

```
#!/bin/bash


echo "Vous avez lancé $0, il y a $# paramètre"
echo "Le paramètre 1 est $1"
echo "l'ensemble des paramètres est $*"
```

```
najihi@ubuntu:~$ vi variable.sh
najihi@ubuntu:~$ chmod u+x variable.sh
najihi@ubuntu:~$ ./variable.sh param1 param2 param3
Vous avez lancé ./variable.sh, il y a 3 paramètre
Le paramètre 1 est param1
l'ensemble des paramètres est param1 param2 param3
```

Substitution

- Substitution de variables
- Substitution de commandes

Substitution de variables:

- ${var:-valeur}

```
najihi@ubuntu:~$ VAR=valeur
najihi@ubuntu:~$ echo ${VAR:-'defaut'}
valeur
najihi@ubuntu:~$ unset VAR
najihi@ubuntu:~$ echo ${VAR:-'defaut'}
defaut
```

- ${variable:?message}

```
najihi@ubuntu:~$ VAR=valeur
najihi@ubuntu:~$ echo ${VAR:?'defaut'}
valeur
najihi@ubuntu:~$ unset VAR
najihi@ubuntu:~$ echo ${VAR:?'defaut'}
bash: VAR: defaut
```

- ${variable:=valeur}

```
najihi@ubuntu:~$ VAR=valeur
najihi@ubuntu:~$ echo ${VAR:='defaut'}
valeur
najihi@ubuntu:~$ unset VAR
najihi@ubuntu:~$ echo ${VAR:='defaut'}
defaut
```

- ${variable:+valeur}

```
najihi@ubuntu:~$ VAR=valeur
najihi@ubuntu:~$ echo ${VAR:+'defaut'}
defaut
najihi@ubuntu:~$ unset VAR
najihi@ubuntu:~$ echo ${VAR:+'defaut'}
```

Substitution de commande

- Syntaxe

$(COMMANDE) ou bien `COMMANDE`

```
najihi@ubuntu:~$ echo "la date du systeme est : $(date)"
la date du systeme est : samedi 12 mai 2012, 00:29:21 (UTC+0200)
najihi@ubuntu:~$ echo "la date du systeme est : `date`"
la date du systeme est : samedi 12 mai 2012, 00:29:51 (UTC+0200)
```

Conclusion